Lab 2: Implementing Adapter Design Pattern

Existing.py

Now let's see an example of adapt design pattern in practice, assume that we already have a class for playing the MP3 in this class MP3 player. This glass provides an interface or function to play that given MP3 file.

```
class MP3player:
    def play(self, filename:str):
        if filename.endswith(".mp3"):
            print("Playing {0}".format(filename))
        else:
            raise Exception("{0} not supported".format(filename))
```

In this class we have a method play, and it accepts the filename, and it checks whether the filename is ending with an MP3 it is the case, then it simply plays.

In our case. I have simply kept a print statement just to denote that this particular MP3 file is being played. This is a dummy implementation of this play method just for the simplicity's sake.

So, this way we have this MP3 class providing one method to play the MP3 file that given an MP3 file.

Now what we want in our application is.

We want a class which is able to play a list of MP3 files, not a single MP3 file, rather, given a list of MP3 files, the class should be able to play all these files one by one. This is our required functionality.

Now, in order to make use of this MP3 class, what we do is we write an adapter. Now let us see this adapter.

So come into this adapter class. What we do is we are writing a new class, a new MP3 player, and in the new MP3 player, class constructor, we set an attribute player, which is an instance of existing MP3 player class.

```
from existing import MP3player
class NewMP3Player:
    def __init__(self, mplayer:MP3player):
        self.player = mplayer
        self.file_list = []

    def add_file(self, file:str):
        if file.endswith(".mp3"):
            self.file_list.append(file)
        else:
            raise Exception("file not supported")

    def play(self):
        for f in self.file_list:
            self.player.play(f)
```

And along with that, we initialize a file list in this new adopter class. new MP3 player class, which is an adapter class, we provide additional interface to add a file to a list. So, this method adds the given file to the list, and we provide a new method play for the application in this play method.

What we do is we pick the files from this list one by one and ask the existing MP3 player class to play that file. So, in this way we provide the required functionality to our client or the application.

Summary

If we see our application, what it does is, first of all, it creates an instance of existing class, then it creates an instance of adaptor given the existing class object.

And after that, it uses the instance of adopter Class to add different files to the list and ask the adopter class object to play all these files.

So, in this way, our application gets the required functionality using the adapter, which internally uses the instance of existing class, right. So now let us try to run this program.

```
from existing import MP3player
from adapter import NewMP3Player

if __name__ == "__main__":
    oPlayer = MP3player()
    player = NewMP3Player(oPlayer)
    player.add_file("a.mp3")
    player.add_file("b.mp3")
    player.add_file("c.mp3")
    player.play()
```

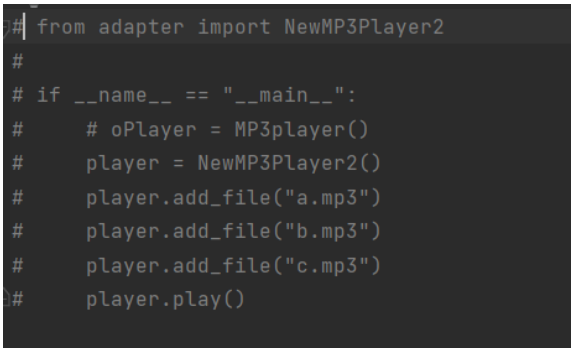
This is an example of object adaptors where our adapter, the new MP3 player object, had an instance of existing class object. So, this is object adapter.

Now, let us see an example of a class adapter, so again, our existing class remains the same. We come to the adapter, and we modify our adapter or something like this. So, in case of class adapter, our adapter does not have any instance of the existing class, rather our adapter class is derived from the existing class. So, if you see here the new MP3player2, it is being derived from MP3 player class.

So, in the constructor of this new MP3 player2 we set a file list, similar to our earlier adopter, we provide a method to add a file into the list and we provide a method to play that list, the MP3 file list.

However, in this play method, what we do is we pick the file from the list one by one. We use the base glass to play these files.

So, this is how our new adapter works now, how we use this new adapter in the application. Let's switch to the main.py.



This is our new application, which imports new MP3 player2. This is the second adapter. You just simply create the instance of adapter class. Now, this time it does not need to create the instance of existing class, or this particular statement is commented.

And after that, it simply adds the file to this adapter object and asked the adapter object to play these files one by one. Right. So, this is a simple application of adapter class. This is the adapter design button, adapter, class design button.

Run this program